Final Exam : Wednesday April 19th, 2000
Instructor : Osmar Zaïane
Section : B1

Student Name :_____

Student ID :_____

Seat Number : _____

- The duration of the exam is 3 hours.
- There are 5 sections worth a total of 100 points.
- This final exam will count for 35% of the overall course grade.
- Read all questions carefully.
- Use a pen and not a pencil.
- For full grades, answer all parts of all questions.
- Be concise and give clear and legible answers in the provided spaces.
- Non-legible answers will not be marked.
- Use the back of the pages for your rough notes and calculations.
- Cheating is a serious offence in the code of student behaviour.
- No books, notes or other aids are permitted during the exam.
- Good luck!

Section 1	Section 2	Section 3	Section 4	Section 5	**Total**
Out of 10	Out of 30	Out of 13	Out of 32	Out of 15	**Out of 100**

Section 1: Simple Java programs [10 points]

1- (2 points) Use the Keyboard class and write a java program that reads a string from the keyboard and displays it on the screen twice on two separate lines.

```
public class question1 {
   /* Reads a string and prints it twice */

      public static void main (String args[]) {

      String aString;
      System.out.print("Please enter a string:");
      aString=Keyboard.in.ReadString();
      System.out.println(aString);
      System.out.println(aString);
      }
}
```

2- (3 points) Use the Keyboard class and write a java program that reads two integers from the keyboard and displays their sum. Ignore input validation.

```
public class question2 {
   /* Reads two integers and prints the sum */

      public static void main (String args[]) {

      Integer firstInteger, secondInteger;
      System.out.println("Please enter first integer:");
      FirstInteger=Keyboard.readInteger();
      System.out.println("Please enter first integer:");
      secondInteger=Keyboard.readInteger();
      System.out.println("sum=" + firstInteger.intValue() +
                                  SecondInteger.intValue());

      }
}
```

3- (5 points) Use the Keyboard class and write a java program that reads one number between 0 and 9 and displays "The number entered was:" followed by the number. The program should validate the input. Only one digit is accepted between 0 and 9.

```
public class question3 {
   /* Validates the input of a digit between 0 and 9 */

      public static void main (String args[]) {

      Integer number;

      do {
            System.out.print("Please enter your choice (0..9):");
            number = Keyboard.in.readInteger();
      } while (number==null ||
               number.intValue() < 0 ||  number.intValue() >9);



      }
}
```

Section 2: Multiple choice [30 points] (2 points each)

Circle exactly ONE choice as the best answer to each question.

1- Which one of these statements is NOT true?

a) The size of a vector changes when an object is added or removed from the vector;
b) A vector is bigger than an array;
c) A vector can contain other vectors;
d) An element cannot be added or removed from an array once the array is created.

2- A, B, and C are variables of type *int*. Which of the following program segments would NOT have the effect of interchanging the values of the variables A and B?

a) C=A; A=B; B=C;
b) C=A; B=A; A=B;
c) A=A+B; B=A-B; A=A-B;
d) C=B; B=A; A=C;

3- Which one of the following would generate a syntax error?

a) int Marks[];
b) int[] Marks;
c) int Marks[] = {1,2,3,4};
d) int Mark[]={0,1,2,3};
e) int Mark[4];

4- How many stars (*) are printed when the following Java code is executed?

```
for(i=4;i<=20;i++)
        System.out.print("*");
```

a) 15
b) 16
c) 17
d) 19
e) 20

5- How many stars (*) are printed when the following Java code is executed?

```
for(i=0;i<6;i++)
    for(j=i;j>=0;j--)
        System.out.print("*");
```

a) 15
b) 21
c) 6
d) 20
e) 16
f) more than the above numbers.

6- How many stars (*) are printed when the following Java code is executed?

```
i=0;
while (i>=0) {
  if (i>5) i=i-7;
  else i=i+1;
   System.out.print("*");
}
```

a) 30
b) 6
c) 7
d) 35
e) more than the above numbers.

7- Sequential search is slower than binary search on an array with N elements because:

a) Sequential search has to select elements;
b) Binary search uses 0 and 1 only;
c) Sequential search makes in the order of N^2 comparisons;
d) Binary search makes in order of $log_2(N)$ comparisons;
e) Arrays can be very big.

8- What is the most likely cause of the error: _exceptionOccurred: java.lang.NullPointerException()?

a) Forgetting to include "throws Exception" in a method declaration;
b) **Sending a message to an object** that was not created and initialized;
c) Trying to put the null object in a vector;
d) Assigning null to an object reference.

9- The following is a static method. What is NOT true about it?

```
static int abc(int n) {
    int a,b;
    a=n/2;
    b=abc(a);
    c=b+2;
    return c;
}
```

a) It is shared by all objects in the class;
b) It is a recursive method;
c) It will generate a syntax error at compile time;
d) **If there were syntax errors and they were fixed, the initial call abc(2) would return 3;**
e) If there were syntax errors and they were fixed, the initial call abc(1000) would repeat indefinitely.

10- After sorting a collection of strings in alphabetical order, the string "memory" would be placed before the string "memorization" because:

a) The string "memory" is shorter than the string "memorization";
b) The order is ascending order;
c) You need memory to memorize;
d) **The string "memorization" is** placed before the string "memory".

11- Protected variables

a) Are shared like static variables;
b) Can not be changed once they have a value assigned to them;
c) **Can be accessed directly in subclass code;**
d) Are instance variables that are not affected by side-effects;
e) Cannot be used as parameters;
f) Do not generate exceptions.

12- Which of the following statements is NOT true?

a) An object from a subclass can receive the messages that an object from the superclass receives.
b) While objects from the subclass and superclass can receive the same messages, their behaviour vis-à-vis these messages can be different.
c) **A superclass has more methods than its subclass.**
d) *super( )* is used to call upon the constructor of the superclass from within the constructor of the subclass.
e) *this( )* is used to call upon another constructor of the same class.

13- Black box testing is a process to:

a) Eliminate syntax errors from Java programs;
b) Validate Java programs;
c) **Make sure methods of a class behave as described in the specifications;**
d) Test the black box class.
e) Describe hypothetical Java classes.
f) Make sure that all paths of the Java code are visited.

14- Given the collection: 5, 10, 13, 15, 21, 25, 32, 45, 52, 55, 65, 90, 93. How many comparisons are done in a binary search that is looking for 95?

a) 3
b) 4
c) 5
d) 12
e) 13

15- Which of the following statements is true?

a) The size of an array can be determined by sending the message size() to the array.
b) The element of a vector can be accessed without sending a message to the vector.
c) Binary search can be used to search for an element in an unsorted collection.
d) Merge-sort works only with arrays.
e) To terminate, a recursive method call, the code of the recursive method must contain at least one case where no recursive method call is made.

Section 3: Merging lists [13 points]

1- [4 points] Given three arrays A, B initialized with some non sorted integers, and C with a size equal to the sizes of A and B put together, you are asked to write a Java program to combine A and B elements into the array C and print all the elements in C. C is **not** supposed to be **sorted**. Fill in the missing Java statements.

```
public class combine {
   /* Combine 2 arrays into another non sorted array */

      public static void main (String args[]) {

           int A[]={23,34,12,5,64,7,91};
           int B[]={45,65,35,70,4,93};
           int C[];
           int k;
           C= new int[A.length+B.length];




            for(k=0;k<A.length;k++) C[k]=A[k];
            for(k=0;k<B.length;k++) C[k+A.length]=B[k];

           for(k=0;k<C.length;k++)
              System.out.print(C[k]+" ");
           System.out.println();
      }
}
```

2- [9 points] Given three arrays A, B initialized with sorted integers, and C with a size equal to the sizes of A and B put together, you are asked to write a Java program to combine A and B elements into the array C such that C is also **sorted**, and print all the elements in C. Fill in the missing Java statements;

```
public class merge {
   /* Combine 2 sorted arrays into another sorted array */

      public static void main (String args[]) {

           int A[]={5, 7, 12, 23,34,64,91};
           int B[]={4, 35, 45,65,70,93};
           int C[];
           int k;
           C= new int[A.length+B.length];









            int i,j;

            i=0; j=0; k=0;

            while (i<A.length && j<B.length) {
                 if (A[i] < B[j]) {
                      C[k]=A[i];
                      k++; i++;
                 } else {
                      C[k]=B[j];
                      k++; j++;
                 }
            }

            while (i<A.length) {
                 C[k]=A[i];
                 k++; i++;
            }

            while (j<B.length) {
                 C[k]=B[j];
                 k++; j++;
            }









           for(k=0;k<C.length;k++)
              System.out.print(C[k]+" ");
           System.out.println();
      }
}
```

Section 4: File input/output, vectors and arrays [32 points]

A friend of yours wants to write a program to keep track of his debts. He has a file containing the list of his friends names, one per line, and has started writing a Java program that reads the file, displays the names one at a time and prompts for the amount overdue, then displays the total debt and writes to another file the names with the amounts due to them. However, he doesn't know Java well enough to finish the program, and asks you to give him a hand by filling in the missing Java statements.

Here is a sample output of the main program. Keyboard input is given in bold:

```
Please enter the input file name: myfriends.txt
File Read.
Please enter the amounts due for each person.
Fred Flintstone >$ 25
John Smith >$ 120.50
Jane Doe >$ 5
Osmar Zaïane >$ 0
Ralph Klein >$ 12
Total debt is $162.50
Please enter the output file name: mydebtlist.txt
File Written.
```

1- [2 points] Fill in the missing Java statements to finish the constructor of the mydebts class.

```
import java.util.*;
import java.io.*;
public class mydebts {
   /* Each instance of this class represents a list of people and
the amount due to them */

/* Instance variables */

Vector peopleNames;
float amount[];

/* constructor */

     public mydebts() {
     /* initializes the vector of names to be empty */


          this.peopleNames = new Vector();



     }
```

2- [6 points] Fill in the missing Java statements to finish the readFile() method that reads the names from the file the name of which is given as argument, and adds the names in the vector. The names are written one per line in the file.

```
public void readFile(String fileName) throws Exception {

/* Reads names from file which name is given and fills my
 vector */

      File aFile;
      FileInputStream inputStream;
      InputStreamReader a Reader;
      BufferReader aBufferReader;

      String aString;
      aFile= new File(fileName);
      inputStream = new FileInputStream(aFile);
      aReader = new InputStreamReader(inputStream);
      aBufferReader = new BufferReader(aReader)

      aString = aBufferReader.readLine();
      while (aString != null) {
           this.peopleNames.addElement(aString);
           aString = aBufferReader.readLine();
      }









      aFile.close();
      System.out.println("File Read.");
}
```

3- [6 points] Fill in the missing Java statements to finish the readDebts() method that reads from the keyboard the amount overdue for each name and stores the amounts in the array of floats. The method reads the debts for every name stored in the vector. The method should prompt for the amount by printing the name of the person like in the sample output.
The Keyboard class has a readFloat() method that reads a float from the keyboard and returns a Float object. The method returns null if the entered data is not valid (not a float). If the user enters an invalid float, assign zero to the corresponding amount.
You can send the floatValue() method to a Float object to get its value in float.

```
public void readDebts() {

/* Reads the debts for every name in the vector and stores
the amounts in my float array at the same position as the
respective position in the vector*/

   this.amount = new float[this.peopleNames.size()];
   System.out.println("Please enter the amount due for each person.");


    int index;
    Float afloat;

    for(i=0; i<this.peopleName.sizes(); i++) {
      aString=(String) this.peopleNames.elementAt(i);
      System.out.print(aString + ">$");
      afloat=Keyboard.in.readFloat();
      if (afloat != null)
           this.amount[i]=aFloat.floatValue();
      else
           this.amount[i]=0.0f;
    }









}
```

4- [3 points] Fill in the missing Java statements to finish the totalDebts() method that calculates and returns the sum of all debts using either an iteration or a recursion approach. **3 extra points will be given for a recursive solution**.

```
public float totalDebts(                    ) {
/* Returns the sum of the debts */
      float sum=0.0f; int i;
      for(i=0;i<this.amount.length;i++) sum+=this.amount[i];
      return sum;
}


public float totalDebts(int index) {
/* Returns the sum of the debts the recursive way*/
      if (index==(this.amount.length-1)
          return this.amount[index]
      return this.amount[index]+totalDebts(index+1);
}
```

5- [6 points] Fill in the missing Java statements to finish the writeFile() method that writes the names and the amounts due to them to the file the name of which is given as argument. Each line in the file contains a name and the corresponding amount, separated by a space.

```
public void writeFile(String fileName) throws Exception {

/* Writes names from vector and amounts from array to file
the name of which is given. */

      File aFile;
      FileOutputStream outputStream;
      aFile= new File(fileName);

      outputStream = new FileOutputStream(aFile);
      PrintStream aPrintStream;
      aPrintStream = new PrintStream(outputStream);

      int i;
      String aString;

      for(i=0; i<this.peopleNames.size(); i++) {
          aString=(String)this.peopleNames.elementAt(i);
          aPrintStream.print(aString);
          aPrintStream.println(" " + this.amount[i]);
      }



      aFile.close();
      System.out.println("File Written.");
}
```

6- [6 points] Write the main program that uses the mydebt class to fulfill your friends needs as described in the sample of output given above. Use totalDebts() method as you defined it and defined its parameters in question 4 (recursive or iterative). For the sake of simplicity, do not consider the number of digits after the decimal point. Just print a float as it is. Note that all necessary local variables for the main method were declared. You do not necessarily need other variables.

```
public class myProgram {

      public static void main (String args[]) {
      /* program statements go here */

    mydebts friends;
    float myTotalDebt;
    String myFileName;



      System.out.print("Please enter the input file name:");
      myFileName=Keyboard.in.readString();

      friends = new mydebts();

      friends.readFile(myFileName);
      friends.readDebts();

      myTotalDebt=friends.totalDebts();
//or myTotalDebt=friends.totalDebts(0); if recursive

      System.out.println("Total debt is $" + myTotalDebt);

      System.out.println("Please enter the output file name:");
      myFileName=Keyboard.in.readString();

      friends.writeFile(myFileName);









      }
}
```

Section 5: Tracing code [15 points]

Consider the following Java classes:

```
public class myQuestion {

/* instance variables */
      protected int a;
      protected int b;

      public myQuestion() {
            this.a = 0;
            this.b = 0;
      }
      public myQuestion(int x, int y) {
            this.a = x;
            this.b = y;
      }
      public myQuestion(int x) {
            this();
            this.a = x;
      }
      public int enquire() {
            return this.a + this.b;
      }
      public int interrogate(int x) {
            return (this.a + this.b)* x;
      }
      public void display() {
            System.out.println(this.a + " " + this.b);
      }
}


public class myProblem extends myQuestion {

/* instance variable */
      protected int c;

      public myProblem() {
            this.c = 0;
      }
      public myProblem(int x, int y, int z) {
            super(y,z);
            this.c = x;
      }
      public myProblem(int x, int y) {
            this();
            super(x,y);
      }
      public myProblem(int x){
            this.c = x;
      }
      public int enquire() {
            this.c = this.a * this.b;
            return this.c;
      }
      public void display() {
            System.out.println(this.a + " " +this.b + " " +this.c);
      }
}
```

a) [3 points] What is the output of the program segment:

```
myQuestion q; q=new myQuestion(1);
System.out.print(q.interrogate(9) + " "); q.display();
```

Output

```
9 1 0
```

b) [3 points] What is the output of the program segment:

```
myQuestion q; q=new myQuestion(20,2);
System.out.print(q.enquire()); q.display();
```

Output

```
22 20 2
```

c) [3 points] What is the output of the program segment:

```
myProblem p; p=new myProblem(20,2);
System.out.println(p.enquire());
System.out.print(p.interrogate(2) + " "); p.display();
```

Output

```
40
44 20 2 40
```

d) [3 points] What is the output of the program segment:

```
myProblem p; p=new myProblem(20,2,4);
p.display();
p.enquire();  p.display();
```

Output

```
2 4 20
2 4 8
```

e) [3 points] What is the output of the program segment:

```
myProblem p; myQuestion q;  p=new myProblem(10); q= new myQuestion(10);
System.out.print(p.enquire() + " "); p.display();
System.out.print(q.enquire() + " "); q.display();
```

Output

```
0 0 0 0
10 10 0
```